برمجة UAC الخاص بويندوز فيستا

بقلم:
محمد سامر أبو سلو

ويضم المواضيع التالية:

UAC Security

تمكين برنامجك من استخدام صلاحيات مدير على فيستا

كيف نقوم بجعل أحد الأزرار في برنامجنا ينفذ أوامر تتطلب صلاحيات مدير في ويندوز فيستا

UAC Security

استعراض UAC

بشكل عام لا يمكن للبرنامج تأدية أعمال تتطلب صلاحيات لا يمتلكها المستخدم فإن لم يمتلك ذلك المستخدم الصلاحيات الكافية لحذف ملفات في مجلد الويندوز فلا يمكن للبرنامج المشغل من قبله أن يحذف تلك الملفات أيضا ومع ذلك يمكن للمستخدم تنفيذ أعمال من المفترض أنه ممنوع منها. والمطورون يعلمون منذ مدة طويلة أن التطبيق يجب أن يمتلك بعض الصلاحيات لكي يتمكن من إتمام العمل فإن كان التطبيق يتطلب العديد من الصلاحيات فوحدهم المستخدمون الذين يمتلكون تلك الصلاحيات يمكنهم تشغيل ذلك البرنامج. ولسوء الحظ فإن العديد من التطبيقات التي تقوم بأعمال قوية تحتاج إلى إنشاء أو حذف ملفات في مجلد الويندوز أو الوصول إلى مناطق متعلقة بالنظام أو التعديل على متغيرات البيئة أو مسجل النظام فإن كان التطبيق يحتاج تلك الصلاحيات فعندها يجب أن يمتلك تلك الصلاحيات عند تشغيله مما يعني أنه على العديد من المستخدمين امتلاك صلاحيات مدير نظام حتى يستطيعوا تشغيل تلك البرامج.

والتعامل مع صلاحيات بهذا المستوى يأتي مع أخطار إضافية فإن أساء التطبيق التصرف فقد يتسبب بانهيار النظام حتى لو كان التطبيق ذات نفسه يعمل بصورة طبيعية فقد يقوم المستخدم بعمل شئ كارثي عن طريق الخطأ عندما يكون قد دخل بصلاحيات مدير فقد يقوم بحذف ملفات هامة يصبح معها من المستحيل استعادة النظام ويكون الحل الأمثل في هذه الحالة هو السماح للتطبيق برفع مستوى الصلاحيات التي يستخدمها بشكل مؤقت أثناء تأديته لتلك الوظائف القوية فإن اخطأ التطبيق عند تشغيله لجزئية معينة من الكود فلن يكون لديه الصلاحيات الكافية لعمل ضرر كبير ولن يكون للمستخدم صلاحيات مدير بشكل دائم وبهذا نكون قد قللنا من احتمال الحوادث المدمرة في النظام.

في نسخ الويندوز السابقة لفيستا عندما تقوم بالدخول كمستخدم يمتلك صلاحيات مدير عندها ستتمكن من القيام بعمل أي شئ تقريبا ولكن في ويندوز فيستا فأن الـ UAC يتصرف بطريقة مختلفة قليلا فعندما تدخل كمدير يكون دخولك عبارة عن شقين الأول عبارة عن مستخدم عادي ذو صلاحيات محدودة والثاني مدير نظام يمتلك كافة الصلاحيات ففي البداية يكون عملك كمستخدم عادي حيث يتم استخدام الشق الثاني عند الحاجة فقط فعندما تريد القيام بعملية تحتاج إلى صلاحيات إضافية فالـ UAC يظهر لك صندوق حوار يسألك الموافقة فإن وافقت على تنفيذ ذلك العمل يتم رفع صلاحياتك بشكل مؤقت إلى مستوى مدير حتى ينتهي تنفيذ ذلك العمل وعندها تعود صلاحياتك إلى مستخدم عادي ثانية وإن كنت قد دخلت باسم مستخدم عادي لا يمتلك صلاحيات مدير فمازال بإمكانك تنفيذ أمر يتطلب تلك الصلاحيات المرتفعة حيث يظهر لك الـ UAC صندوق حوار تحذيري يمكنك من الدخول كمدير فإن قمت بالدخول كمدير بنجاح عندها يتم منحك صلاحيات مدير بشكل مؤقت حتى ينتهي تنفيذ ذلك العمل. ويكون الفرق بين الحالتين بسيطا فعندما تدخل كمدير فإن الـ UAC يسألك موافقتك على العمل بالصلاحيات المرتفعة وإن دخلت كمستخدم آخر فإن الـUAC يطلب منك إدخال كلمة السر الخاصة بالمدير

التصميم من اجل UAC

لن يقوم الـ UAC برفع صلاحيات التطبيق بعد أن تم تشغيله فهو يقوم بإسناد الصلاحيات لذلك التطبيق عندما يبدأ ولن يقوم بعدها بتغيير تلك الصلاحيات فإن احتاج التطبيق للعمل بصلاحيات مرتفعة فعليه أن يحصل على تلك الصلاحيات عندما يبدأ ولتجنب إعطاء التطبيق صلاحيات أكثر من اللازم يجب عليك تقسيم كودك إلى أجزاء بحسب احتياجه لتلك الصلاحيات فالبرنامج الرئيسي يجب أن يعمل بصلاحيات عادية ولاحقا يجب عليه تنفيذ تطبيقات أخرى تعمل بصلاحيات مرتفعة عند الحاجة.

فمثلا إن كان لدينا تطبيق يقوم بحفظ البيانات في قاعدة بيانات Sql Server فهو لا يحتاج لصلاحيات مدير ومع ذلك إن أراد إنشاء ملف بملخص العمليات في مجلد الويندوز – مجلد محمي – فسيحتاج عندها لصلاحيات مدير فيمكنك عندها تقسيم التطبيق إلى عدة أجزاء فالتطبيق الرئيسي يقوم بمعظم العمل وتطبيق آخر يقوم بكتابة معلومات الخلاصة إلى ذلك الملف عندها يمكن للتطبيق الأول تشغيل الثاني من أجل كتابة المعلومات في ذلك الملف.

وعندما يكون بالإمكان يفضل أن تعيد كتابة التطبيق لتجنب استخدام صلاحيات مرتفعة فالعديد من البرامج على سبيل المثال تكون منصبة في المجلد Program Files وهذا من المجلدات المحمية وبهذا إن احتاج التطبيق إلى تخزين معلومات في ملف متواجد بنفس المجلد الذي يحتوي على الملف التنفيذي للتطبيق فسوف يحتاج إلى صلاحيات إضافية للقيام بتلك العملية ويمكنك تجاوز هذه المشكلة بجعل التطبيق يكتب ذلك الملف في المجلد الخاص بالمستخدم الحالي. والعمليات الأخرى التي تحتاج لصلاحيات مرتفعة تتضمن الكتابة إلى المجلدات المحمية والتعامل بشكل مباشر مع العتاد وتعديل الأقسام المحمية في سجل النظام مثل HKEY_LOCAL_MACHINE.

وتقسيم التطبيق إلى أقسام تتطلب صلاحيات مرتفعة وأخرى لا تتطلب تلك الصلاحيات لا يمكن التطبيق من استخدام أقل الصلاحيات الممكنة فحسب ولكنه يبسط القسم الأخطر في كودك ويجعله أسهل للتنقيح فمثلا يمكننا استخدام كود شبيه بالتالي لتنفيذ تطبيق يتطلب صلاحيات مرتفعة

```
Private Sub btnRun_Click() Handles btnRun.Click
   Try
      ' Start the process.
      Dim pro As System.Diagnostics.Process
      pro = System.Diagnostics.Process.Start( _
         txtProgram.Text, txtArguments.Text)
      ' Wait for the process to exit.
      pro.WaitForExit()
      ' Display the process's exit code.
      MessageBox.Show("Exit code: " & pro.ExitCode)
   Catch ex As System.ComponentModel.Win32Exception
      ' This happens if the user fails to elevate to Administrator.
      MessageBox.Show("Operation canceled", _
         "Canceled", MessageBoxButtons.OK, _
         MessageBoxIcon.Information)
   End Try
End Sub
```

الكود السابق يستخدم الوظيفة System.Diagnostics.Process.Start لتشغيل التطبيق ممررا مسار التطبيق الذي نريد تنفيذه ومحددات سطر الأوامر الخاصة بها وهو يستخدم الدالة WaitForExit من الغرض المعاد التي تنتظر حتى الانتهاء من تنفيذ البرنامج ثم يتم التأكد عبر الخاصية ExitCode من القيمة المعادة من التطبيق المنفذ.

ويمثل الكود التالي الإجراء main في البرنامج المستدعى

```
Function Main(ByVal cmdArgs() As String) As Integer
   Dim frm As New frmChoices
   ' Display the arguments.
   For Each str As String In cmdArgs
      frm.lstArguments.Items.Add(str)
   Next str
   ' Select the first item.
   If frm.lstArguments.Items.Count > 0 Then
      frm.lstArguments.SelectedIndex = 0
   End If
   ' Return the index of the selected item.
   If frm.ShowDialog() = DialogResult.Cancel Then
      Return -1
   Else
      Return frm.lstArguments.SelectedIndex
   End If
End Function
```

حيث يبدأ التطبيق بإنشاء نموذج frmChoices وإضافة محددات سطر الأوامر إلى صندوق القائمة lstArguments ونختار العنصر الأول منه ثم نظهر النموذج فإن قام المستخدم بالضغط على الزر Cancel فالتطبيق يعيد القيمة -1 وإن ضغط على الزر OK فهو يعيد قيمة الخاصية Index من صندوق القائمة والموافقة للعنصر الذي تم اختياره منها والكود المستدعي للتطبيق يستقبل تلك القيمة عبر الخاصية ExitCode.

وكجزء من خصائص المستخدم لـ UAC فأي عمل يتطلب صلاحيات مرتفعة يجب أن يتم تعليمه بواسطة الدرع القياسي لـ UAC حيث يجب إظهاره لتحذير المستخدم بأنه ينفذ تطبيق يتطلب صلاحيات مرتفعة. وفي الوقت الحالي لا توجد طريقة بسيطة لإظهار ذلك الدرع في فيجول بايزيك لذلك سنستخدم دالات API لجعل الزر يظهر ذلك الدرع كما هو ظاهر في قطعة الكود التالية

```
Imports System.Runtime.InteropServices

Module UacStuff
   Declare Auto Function SendMessage Lib "user32.dll" _
```

```
  (ByVal hWnd As HandleRef, ByVal msg As Int32, _
  ByVal wParam As IntPtr, ByVal lParam As IntPtr) As Int32

 ' Make the button display the UAC shield.
 Public Sub AddShieldToButton(ByVal btn As Button)
   Const BCM_SETSHIELD As Int32 = & H160C
   btn.FlatStyle = Windows.Forms.FlatStyle.System
   SendMessage(New HandleRef(btn, btn.Handle), _
   BCM_SETSHIELD, IntPtr.Zero, CType(1, IntPtr))
 End Sub
End Module
```

ففي البداية نقوم بتعريف الدالة SendMessage المتواجدة في المكتبة User32.dll حيث يقوم الإجراء AddShieldToButton بضبط الخاصية FlatStyle الخاصة بالزر إلى System ثم يستخدم الدالة SendMessage لإرسال الرسالة BCM_SETSHIELD إلى الزر ولا توفر لنا مايكروسوفت حاليا طريقة لإضافة الدرع لتحكمات أخرى غير زر الأوامر فإن أردت إضافته إلى تحكم أخر فستقوم بذلك لوحدك كما يمكنك عمل صورة للدرع ووضعها ببساطة على تحكماتك ولكن هذه الصورة لن تتغير إن تم تغيير صورة الدرع الخاصة بالنظام

رفع صلاحيات البرامج

يمكن للمستخدم رفع المستوى الذي يتم تنفيذ التطبيق ضمنه بواسطة اختيار الأمر Run As Administrator من القائمة التي تظهر لك عند الضغط بزر الفأرة اليميني على الملف التنفيذي للتطبيق فيقوم النظام بإظهار صندوق حوار UAC الخاص وبعد أن يقوم المستخدم بإدخال كلمة سر المدير يتم تنفيذ البرنامج باستخدام الصلاحيات المرتفعة وهذه الطريقة بسيطة ولا تتطلب تدخلا منك كمبرمج ولكنها تتطلب من المستخدم القيام بخطوة إضافية ولهذا قد لا تكون هذه الفكرة هي الحل الأفضل دوما.

ويمكننا جعل تطبيقنا يبدأ تطبيق معين باستخدام صلاحيات مرتفعة بطريقة تشابه تلك الطريقة التي يستخدمها المستخدم فهو يبدأ تشغيل التطبيق طالبا من النظام تشغيله بصلاحيات مدير حيث يمكن استخدام كود شبيه بالتالي لتشغيل تطبيق آخر بصلاحيات مرتفعة مع أن ذلك التطبيق بذاته لا يطلب تلك الصلاحيات عند بدء تشغيله

```
Try
 ' Use the runas verb to start the process.
 Dim psi As New ProcessStartInfo
 psi.Verb = "runas"
 psi.UseShellExecute = True
 psi.FileName = txtProgram.Text
 psi.Arguments = txtArguments.Text
 Dim pro As System.Diagnostics.Process
 pro = System.Diagnostics.Process.Start(psi)
 ' Wait for the process to exit.
 pro.WaitForExit()
 ' Display the process's exit code.
 MessageBox.Show("Exit code: " & pro.ExitCode)
Catch ex As System.ComponentModel.Win32Exception
 ' This happens if the user fails to elevate to Administrator.
 MessageBox.Show("Operation canceled", _
   "Canceled", MessageBoxButtons.OK, _
   MessageBoxIcon.Information)
End Try
```

حيث يبني الكود السابق الغرض ProcessStartInfo واصفا التطبيق الذي سيشغله الكود حيث يقوم بضبط الخاصية Verb إلى القيمة runas لكي يبين للنظام أن التطبيق يجب أن يتم تشغيله كمدير كما يضبط اسم البرنامج ومحددات بدء التشغيل الخاصة به.

وإن كنت تعلم أن التطبيق يجب أن يتم تشغيله دوما باستخدام صلاحيات مرتفعة يمكنك جعل ذلك التطبيق يطلب رفع صلاحياته بنفسه وذلك باستخدام manifest مضمنة داخل الملف التنفيذي للتطبيق ولإنشائها انقر نقرا مزدوجا على My Project في Solution

وفي صفحة Application انقر على الزر View UAC Settings الذي يقوم بفتح الملف app.manifest حيث يظهر الكود التالي المحتويات الابتدائية لذلك الملف

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1"
xmlns:asmv1="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-microsoft-
com:asm.v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <!-- UAC Manifest Options
            If you want to change the Windows User Account Control level replace the
            requestedExecutionLevel node with one of the following.

        <requestedExecutionLevel  level="asInvoker" uiAccess="false" />
        <requestedExecutionLevel  level="requireAdministrator" uiAccess="false" />
        <requestedExecutionLevel  level="highestAvailable" uiAccess="false" />

            If you want to utilize File and Registry Virtualization for backward
            compatibility then delete the requestedExecutionLevel node.
        -->
        <requestedExecutionLevel level="asInvoker" uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
</asmv1:assembly>
```

ولجعل التطبيق يطلب من UAC رفع صلاحياته غير قيمة السطر requestExecutionLevel إلى requireAdministrator والآن عندما تقوم بعمل Compile للتطبيق يقوم فيجول ستوديو بتعليم التطبيق بأنه بحاجة إلى صلاحيات مدير فعندما يقوم المستخدم أو أي برنامج آخر بتشغيله سيحاول النظام بصورة آلية رفع صلاحياته مظهرا صندوق الحوار الخاص بـ UAC للمستخدم طالبا منه الموافقة على رفع تلك الصلاحيات

الخلاصة

القواعد الأساسية لبرمجة UAC تتطلب استخدام الحد الأدنى من الصلاحيات لتنفيذ العمل المراد ويجب على التطبيق استخدام صلاحيات مستخدم عادي عندما يكون ذلك ممكنا وإن كان عليه تنفيذ مهمة تتطلب صلاحيات أكبر فيجب عليه تنفيذ تطبيق آخر منفصل يمتلك تلك الصلاحيات المرتفعة.

وقد ورد في هذه المقالة ثلاثة طرق لبدء البرنامج بصلاحيات مرتفعة: الأولى هي الطلب من المستخدم فعل ذلك وذلك من خلال النقر بزر الفأرة اليميني على التطبيق واختيار الأمر Run As Administrator وهذه ليست بالطريقة الملائمة بشكل عام ولكنها تبقى مقبولة إن كان المستخدم سيشغل ذلك التطبيق مرات نادرة والثانية هي جعل التطبيق يبدأ التطبيق الآخر بصلاحيات مرتفعة وهذه طريقة أفضل من الأولى ولكنه مازال بالإمكان تشغيل التطبيق بدون الصلاحيات التي يحتاجها والثالثة هي تضمين manifest ضمن التطبيق المستدعى لجعله يطلب صلاحيات مرتفعة في كل مرة يبدأ فيها تشغيله

تمكين برنامجك من استخدام صلاحيات مدير على فيستا

- لتمكين برنامجك من العمل بصلاحيات مدير شغل بيئة التطوير دوما بصلاحيات مدير - انقر بزر الفأرة اليميني على اختصار بيئة التطوير واختر الأمر Run As Administrator -
- افتح خصائص My Project ثم انقر زر View UAC Settings من صفحة Application ثم في نافذة خصائص UAC التي تظهر لك استبدل السطر

```
<requestedExecutionLevel level="asInvoker" uiAccess="false" />
```

- بالسطر

```
<requestedExecutionLevel  level="requireAdministrator" uiAccess="false" />
```

- نفذ الأمر Build Solution من قائمة Build وبيئة التطوير مازالت تعمل ضمن مستوى Administrator كما تأكد بأنك تستخدم بيئة التطوير بصلاحيات مدير عندما تقوم بعمل برنامج الـ Setup أيضا

كيف نقوم بجعل أحد الأزرار في برنامجنا ينفذ أوامر تتطلب صلاحيات مدير في ويندوز فيستا

نحتاج في بعض الأحيان للقيام بأعمال تتطلب صلاحيات خاصة في ويندوز فيستا وهنا سنواجه منعا من قبل UAC الخاص بويندوز فيستا ولكي يتمكن برنامجنا من تنفيذ هذه المهمة يجب علينا تنفيذ ذلك الكود بمستوى صلاحيات مدير Administrator حيث سنقوم في البداية بتعريف فئة تتعامل مع نظام الأمان في ويندوز فيستا مستخدمين الفئة WindowsIdentity للتعرف على مستخدم ويندوز الذي نعمل عليه والفئة WindowsPrincipal للتعرف على المجموعات التي ينتسب إليها ذلك المستخدم ثم نتحقق من أنه يعمل بصلاحيات مدير كما في الإجراء

```
Friend Shared Function IsAdmin() As Boolean
    Dim id As WindowsIdentity = WindowsIdentity.GetCurrent()
    Dim p As WindowsPrincipal = New WindowsPrincipal(id)
    Return p.IsInRole(WindowsBuiltInRole.Administrator)
End Function
```

فإن لم يكن المستخدم يعمل بصلاحيات مدير هنا نعيد بدء العملية الحالية Restart Current Process رافعين مستوى صلاحيات المستخدم إلى مستوى مدير كما في الإجراء

```
Friend Shared Sub RestartElevated()
    Dim startInfo As ProcessStartInfo = New ProcessStartInfo()
    startInfo.UseShellExecute = True
    startInfo.WorkingDirectory = Environment.CurrentDirectory
    startInfo.FileName = Application.ExecutablePath
    startInfo.Verb = "runas"
    Try
        Dim p As Process = Process.Start(startInfo)
    Catch ex As System.ComponentModel.Win32Exception
        Return 'If cancelled, do nothing
    End Try

    Application.Exit()
End Sub
```

بقي لدينا إضافة أيقونة الدرع الخاصة بالأزرار التي تستخدم صلاحيات مدير إلى زر الأوامر المطلوب حيث يتم ذلك باستخدام الدالة SendMessage الموجودة في المكتبة user32.dll التي تقوم بإرسال الرسالة BCM_SETSHIELD إلى الزر المطلوب كما في الإجراء

```
Friend Shared Sub AddShieldToButton(ByVal b As Button)
    b.FlatStyle = FlatStyle.System
    SendMessage(b.Handle, BCM_SETSHIELD, 0, &HFFFFFFFF)
End Sub
```

وسيصبح الكود الكامل للفئة التي سنستخدمها لإجراء عملية تمكين الزر من تنفيذ أعمال تتطلب صلاحيات مدير كما يلي

```
Imports System
Imports System.Collections.Generic
Imports System.Text
Imports System.Runtime.InteropServices
Imports System.Diagnostics
Imports System.Windows.Forms
Imports System.Security.Principal

Public Class VistaSecurity
    Private Declare Auto Function SendMessage Lib "user32.dll" _
        (ByVal HWND As IntPtr, ByVal MSG As UInteger, ByVal WParam As UInt32, _
         ByVal LParam As UInt32) As UInt32

    Private Const BCM_FIRST = &H1600
    Private Const BCM_SETSHIELD = (BCM_FIRST + &HC)

    Friend Shared Function IsVistaOrHigher() As Boolean
```

```
        Return Environment.OSVersion.Version.Major < 6
    End Function

    '/ <summary>
    '/ Checks if the process is elevated
    '/ </summary>
    '/ <returns>If is elevated</returns>
    Friend Shared Function IsAdmin() As Boolean
        Dim id As WindowsIdentity = WindowsIdentity.GetCurrent()
        Dim p As WindowsPrincipal = New WindowsPrincipal(id)
        Return p.IsInRole(WindowsBuiltInRole.Administrator)
    End Function

    '/ <summary>
    '/ Add a shield icon to a button
    '/ </summary>
    '/ <param name="b">The button</param>
    Friend Shared Sub AddShieldToButton(ByVal b As Button)
        b.FlatStyle = FlatStyle.System
        SendMessage(b.Handle, BCM_SETSHIELD, 0, &HFFFFFFFF)
    End Sub

    '/ <summary>
    '/ Restart the current process with administrator credentials
    '/ </summary>
    Friend Shared Sub RestartElevated()
        Dim startInfo As ProcessStartInfo = New ProcessStartInfo()
        startInfo.UseShellExecute = True
        startInfo.WorkingDirectory = Environment.CurrentDirectory
        startInfo.FileName = Application.ExecutablePath
        startInfo.Verb = "runas"
        Try
            Dim p As Process = Process.Start(startInfo)
        Catch ex As System.ComponentModel.Win32Exception
            Return 'If cancelled, do nothing
        End Try

        Application.Exit()
    End Sub
End Class
```

دعنا نجرب معا الفئة التي قمنا بإنشائها للتو

سأقوم بالتجربة على كود مقتطف من برنامج قديم لي وهو يراقب خدمة النظام الخاصة بـ SQl Server ويتحكم بها وبما أن كود بدء أو إيقاف هذه الخدمة يعتبر من الأمور التي تحتاج إلى صلاحيات مدير لذا سأضع فقط قطعة الكود التي تفيدنا هنا حيث سنحتاج في البداية إلى إضافة مرجع إلى System.ServiceProcess وإلى الاستيرادات التالية في بداية الملف أيضا

```
Imports System.ServiceProcess
Imports Microsoft.Win32
```

وهذا هو الكود

```
Private SqlServiceCon As New _
        System.ServiceProcess.ServiceController("MSSQL$SQLEXPRESS")

Private Sub StopSQL()
    Try
        SqlServiceCon.Refresh()
        If SqlServiceCon.CanStop = True Then SqlServiceCon.Stop()

    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

```
Private Sub StartSql()
    Try
        SqlServiceCon.Refresh()
        If SqlServiceCon.Status <> ServiceControllerStatus.Running And _
           SqlServiceCon.Status <> ServiceControllerStatus.StartPending Then

            SqlServiceCon.Start()
        End If
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

الآن سنستخدم فئتنا السابقة VistaSecurity للتحقق أولا من أن برنامجنا يمتلك الصلاحيات المطلوبة باستخدام الدالة IsAdmin فإن لم يمتلك تلك الصلاحيات نعيد بدء العملية Process الحالية رافعين الصلاحيات للمستوى المطلوب باستخدام الدالة RestartElevated كما في الكود

```
If VistaSecurity.IsAdmin = True Then
    StartSql()
Else
    VistaSecurity.RestartElevated()
End If
```

وعملية إضافة أيقونة الدرع إلى زر الأوامر تتم باستخدام الكود

```
VistaSecurity.AddShieldToButton(Button1)
```

وفيما يلي سرد كامل لكود النافذة Form1 التي استخدمناها هنا وهي تمتلك زري أوامر Button1 و Button2

```
Imports System.ServiceProcess
Imports Microsoft.Win32

Public Class Form1

    Private SqlServiceCon As New
System.ServiceProcess.ServiceController("MSSQL$SQLEXPRESS")

    Private Sub StopSQL()
        Try
            SqlServiceCon.Refresh()
            If SqlServiceCon.CanStop = True Then SqlServiceCon.Stop()

        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub StartSql()
        Try
            SqlServiceCon.Refresh()
            If SqlServiceCon.Status <> ServiceControllerStatus.Running And _
               SqlServiceCon.Status <> ServiceControllerStatus.StartPending Then

                SqlServiceCon.Start()
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
```

```
        If VistaSecurity.IsAdmin = True Then
            StartSql()
        Else
            VistaSecurity.RestartElevated()
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles Button2.Click

        If VistaSecurity.IsAdmin = True Then
            StopSQL()
        Else
            VistaSecurity.RestartElevated()
        End If
    End Sub

    Private Sub Form1_Load(ByVal sender As Object, _
                 ByVal e As System.EventArgs) Handles Me.Load

        VistaSecurity.AddShieldToButton(Button1)
        VistaSecurity.AddShieldToButton(Button2)
    End Sub

End Class
```

سؤال

هل من الممكن شرح استخدام هذه الطريقة مع Windows XP

الجواب

هذه الطريقة خاصة حصرا لـ Windows Vista ولا يمكن استعمالها مع Windows XP